4-16-2012

Modeling and Verification of a Dual Chamber Implantable Pacemaker

Zhihao Jiang
University of Pennsylvania, zhihaoj@seas.upenn.edu

Miroslav Pajic
University of Pennsylvania, pajic@seas.upenn.edu

Salar Moarref
University of Pennsylvania, moarref@seas.upenn.edu

Rajeev Alur
University of Pennsylvania, alur@cis.upenn.edu

Rahul Mangharam
University of Pennsylvania, rahulm@seas.upenn.edu

Recommended Citation

Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, and Rahul Mangharam, "Modeling and Verification of a Dual Chamber Implantable Pacemaker", *Lecture Notes in Computer Science: Tools and Algorithms for the Construction and Analysis of Systems* 7214, 188-203. April 2012. http://dx.doi.org/10.1007/978-3-642-28756-5_14

From the 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 – April 1, 2012.

Modeling and Verification of a Dual Chamber Implantable Pacemaker

Abstract
The design and implementation of software for medical devices is challenging due to their rapidly increasing functionality and the tight coupling of computation, control, and communication. The safety-critical nature and the lack of existing industry standards for verification, make this an ideal domain for exploring applications of formal modeling and analysis. In this paper, we use a dual chamber implantable pacemaker as a case study for modeling and verification of control algorithms for medical devices in UPPAAL. We present detailed models of different components of the pacemaker based on the algorithm descriptions from Boston Scientific. We formalize basic safety requirements based on specifications from Boston Scientific as well as additional physiological knowledge. The most critical potential safety violation for a pacemaker is that it may lead the closed-loop system into an undesirable pattern (for example, Tachycardia). Modern pacemakers are implemented with termination algorithms to prevent such conditions. We show how to identify these conditions and check correctness of corresponding termination algorithms by augmenting the basic models with monitors for detecting undesirable patterns. Along with emerging tools for code generation from UPPAAL models, this effort enables model driven design and certification of software for medical devices.

Keywords
Medical Devices, Implantable Pacemaker, Software Verification, Cyber-Physical Systems

Disciplines
Computer Engineering

Comments
From the 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 – April 1, 2012.

Modeling and Verification of a Dual Chamber Implantable Pacemaker

Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, Rahul Mangharam

University of Pennsylvania, Philadelphia PA, USA

Abstract. The design and implementation of software for medical devices is challenging due to their rapidly increasing functionality and the tight coupling of computation, control, and communication. The safety-critical nature and the lack of existing industry standards for verification, make this an ideal domain for exploring applications of formal modeling and analysis. In this paper, we use a dual chamber implantable pacemaker as a case study for modeling and verification of control algorithms for medical devices in UPPAAL. We present detailed models of different components of the pacemaker based on the algorithm descriptions from Boston Scientific. We formalize basic safety requirements based on specifications from Boston Scientific as well as additional physiological knowledge. The most critical potential safety violation for a pacemaker is that it may lead the closed-loop system into an undesirable pattern (for example, Tachycardia). Modern pacemakers are implemented with termination algorithms to prevent such conditions. We show how to identify these conditions and check correctness of corresponding termination algorithms by augmenting the basic models with monitors for detecting undesirable patterns. Along with emerging tools for code generation from UPPAAL models, this effort enables model driven design and certification of software for medical devices.

Keywords: Medical Devices, Implantable Pacemaker, Software Verification, Cyber-Physical Systems

1 Introduction

Over the past four decades, cardiac rhythm management devices such as pacemakers have expanded their role from "keeping the patient alive" to "making the patient's life comfortable". The addition of more safety and efficacy features has resulted in increased complexity, inevitably leading to more safety violations. From 1996-2006, the percentage of software-related causes in medical device recalls have grown from 10% to 21% [1]. During the first half of 2010, the US Food and Drug Administration (FDA) issued 23 recalls of defective devices, all of which are categorized as *Class I*, meaning there is a "reasonable probability that use of these products will cause serious adverse health consequences or death." At least six of the recalls were caused by software defects [2]. As a result, there is a pressing need for standards and tools to certify and verify the safety of software in medical devices. Unlike other industries such as aviation and automotive, the safety concern in the medical device domain is focused on the physical plant, the patient in this case, rather than the controller. As a result, although in aviation and automotive industries, standards are enforced during software development, manufacturing, and post-market change [3, 4], there are no well-established standards for development of software for medical devices. This has prompted recent interest in applying formal modeling and verification techniques to the domain of medical devices [5, 6].

In [7], we proposed a framework to test and validate an Implantable Cardiac Device. In this paper, we design a basic dual chamber pacemaker (DDD) in the model checker UPPAAL [8] and verify it against a set of basic safety requirements. The most critical safety violation for

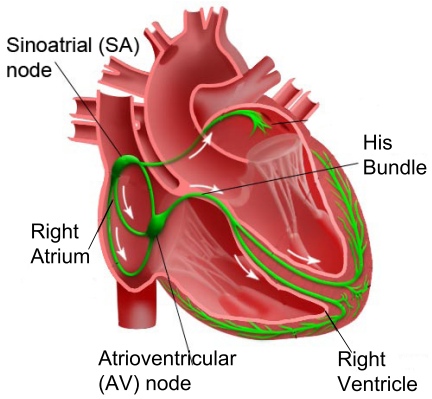


Fig. 1. Electrical Conduction System of the heart

a pacemaker occurs when the pacemaker drives a normal heart into an undesirable condition. The pacemaker software incorporates correction algorithms that are designed to terminate such behaviors. In this paper we discuss two important illustrative cases that are classified as "Pacemaker Mediated Tachycardia (PMT)". We build formal models of the two anti-PMT correction algorithms in UPPAAL, and verify that they terminate the undesirable conditions as intended. Then we verify whether the pacemaker, augmented with the anti-PMT algorithms, still satisfies the safety properties. A similar approach can be used to model and verify other advanced pacemaker algorithms.

Our models and specifications are designed based on descriptions available from Boston Scientific [9, 10], a leading manufacturer of pacemakers, and on extensive medical literature on this topic. At each step of the modeling and verification process, we discuss and justify the choice of heart model, and in particular, we argue that a more physiologically-relevant heart model is needed for more complex properties. The UPPAAL model developed in this paper is freely available online [11]. We hope that these models can be used as a starting point for many purposes (e.g. to build models with costs and probabilities for quantitative analysis). In particular, the verified pacemaker model can be translated into Stateflow charts in Simulink for test generation and code generation [12].

The paper is organized as follows: In Section 2, we introduce the basics about the heart and pacemaker. Section 3 presents UPPAAL models of the basic DDD pacemaker and the heart. In Section 4, we specify two core safety properties that need to hold for all pacemakers, and verify the basic pacemaker model against them. In Section 5, we describe two cases where the pacemaker drives a healthy heart into undesirable states. For each case, we show the existence of the undesirable behaviors using existential queries in UPPAAL and verify whether the corresponding correction algorithms eliminate these behaviors.

2 Heart and Pacemaker Basics

The coordinated contraction of the heart is governed by its Electrical Conduction System (see Fig. 1). The Sinoatrial (SA) node, which is a collection of specialized tissue at the top of the right atrium, periodically generates electrical pulses that can cause muscle contraction. The SA node is controlled by the nervous system and acts as the primary and natural pacemaker of the heart. The electrical pulses first cause both atria to contract, forcing the blood into the ventricles. The electrical conduction is then delayed at the Atrioventricular (AV) node, allowing the ventricles to fill fully. Finally the fast-conducting His-Pukinje system spreads the electrical

activation within both ventricles, causing simultaneous contraction of the ventricular muscles, and pumps the blood out of the heart.

Due to aging and/or diseases, the conduction properties of heart tissue in the electrical conduction system may change. These changes may cause timing anomalies in heart rhythm, thus decrease the blood pumping efficiency of the heart. These timing anomalies are referred to as arrhythmias, and are categorized into *Tachycardia* and *Bradycardia*. Tachycardia features undesirable fast heart rate which impairs hemodynamics. Bradycardia features slow heart rate which results in insufficient blood supply. Bradycardia maybe due to failure of impulse generation with anomalies in the SA node, or failure of impulse propagation where the conduction from atria to the ventricles is delayed or blocked.

Since the heart tissue can be activated by external electrical pulses, Bradycardia can be treated by providing electrical pulses when the heart rate is low. *Implantable Pacemakers* have been developed to deliver timely electrical pulses to the heart to maintain an appropriate heart rate and Atrial-Ventricular synchrony. Implantable pacemakers normally have two leads fixed on the wall of the right atrium and the right ventricle. Activation of local tissue is sensed by the leads, triggering Atrial Sense (AS) and Ventricular Sense (VS) events. Atrial Pacing (AP) and Ventricular Pacing (VP) are delivered if no sensed events occur within deadlines.

In order to deal with different heart conditions, modern pacemakers are able to operate in different modes. The modes are labeled using a three character system. The first character describes the pacing locations, the second character describes the sensing locations, and the third character describes how the pacemaker software responds to sensing. In this work we describe the most commonly used mode of pacemaker, the DDD mode that paces both the atrium and the ventricle, senses both chambers, and sensing can both activate or inhibit further pacing. Similarly, the VDI mode paces only in the ventricle, senses both chambers, and inhibits pacing if event is sensed. [13]

3 System Modeling

3.1 Timed Automata and UPPAAL

Timed automaton [14] is an extension of a finite automaton with a finite set of real-valued clocks. It has been used for modeling and verifying systems which are triggered by events and have timing constraints between events. From the Boston Scientific pacemaker specification [9], the pacemaker can be modeled using this Extended Timed Automata notation, which is a subset of formal semantics in UPPAAL. UPPAAL ([8, 15]) is a standard tool for modeling and verification of real-time systems, based on networks of timed automata. The graphical and text-based interface makes modeling more intuitive. Requirements can be specified using Computational Tree Logic (CTL) [16] and violations can be visualized in the simulation environment.

3.2 System Overview

We modeled both the heart and the pacemaker in UPPAAL. The overview of the closed-loop system is showed in Fig. 2(a). The heart and the pacemaker communicate with each other using broadcast channels. The heart generates *Aget!* and *Vget!* actions, representing atrial and ventricular events that the pacemaker take as inputs. The pacemaker processes the signals and generates pacing actions *AP!* and *VP!* to the corresponding components in the heart.

3.3 Basic DDD pacemaker modeling

The DDD pacemaker has 5 basic timing cycles, as shown in Fig. 2(b). We correspondingly decomposed our pacemaker model into 5 components which correspond to the 5 timers. These

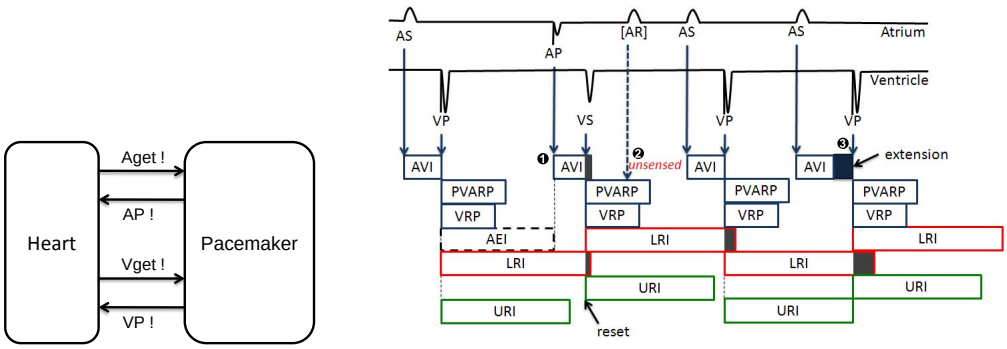


Fig. 2. (a) System Overview, (b) Basic 5 timing cycles of DDD pacemaker

components communicate with each other using broadcast channels and shared variables. The pacemaker design is shown in Fig. 3.

Lower Rate Interval (LRI): This component keeps the heart rate above a minimum value. The LRI automation models the basic timing cycle which defines the longest interval between two ventricular events. The clock is reset when a ventricular event (VS, VP) is received. If no atrial event has been sensed (AS), the component will deliver atrial pacing (AP) after TLRI-TAVI. The UPPAAL design of LRI component is shown in Fig. 3 (a).

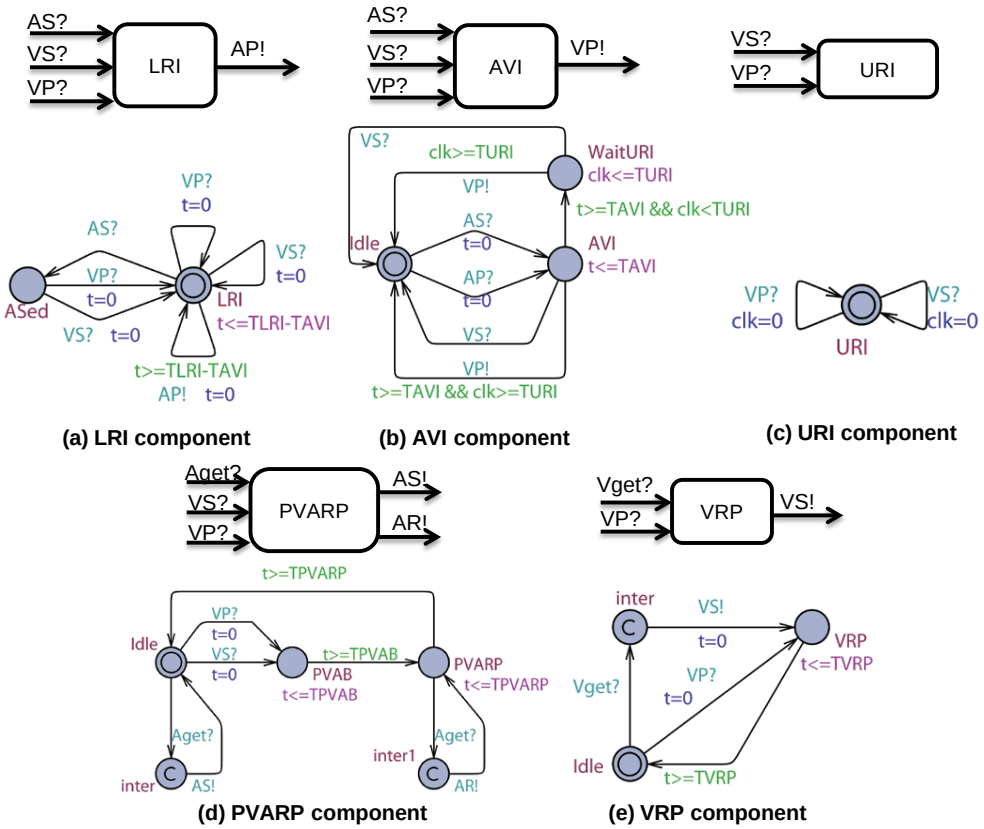


Fig. 3. Components of the pacemaker model in UPPAAL

Atrio-Ventricular Interval (AVI) and Upper Rate Interval (URI): The function of the AVI component is to maintain synchrony between the atria and the ventricles. It defines the longest interval between an atrial event and a ventricular event. If no ventricular event has been sensed (VS) within TAVI after an atrial event (AS, AP), the component will deliver ventricular pacing (VP). In order to prevent the pacemaker from pacing the ventricle too fast, a URI component uses a global clock *clk* to track the time after a ventricular event (VS, VP). The URI automation limits the ventricular pacing rate by enforcing a lower bound on the times between consecutive ventricle events. If the global clock value is less than TURI when the AVI component is about to deliver VP, AVI component will hold VP and deliver it after the global clock reaches TURI. The UPPAAL design of AVI and URI component is shown in Fig. 3 (b) and (c).

Post Ventricular Atrial Refractory Period (PVARP) and Post Ventricular Atrial Blanking (PVAB): The PVARP and the PVAB are initialized by ventricular events. Atrial events during PVAB are ignored and atrial events during PVARP trigger AR event which can be used in some advanced algorithms. The UPPAAL design of PVARP component is shown in Fig. 3 (d).

Ventricular Refractory Period (VRP): Along with the PVARP and PVAB, the VRP is are used to filter noise and early events which could otherwise cause undesired pacemaker behavior. For physiological reasons, there is a blanking interval after each ventricular event that no ventricular event can happen. In our pacemaker model, a VRP period follows each ventricular event (VS, VP) in order to filter noise. The UPPAAL design of VRP component is shown in Fig. 3 (e).

Parameter Selection: In this model, values for parameters are fixed so the model is only one instance of a DDD pacemaker. The values we choose for the parameters in this pacemaker model are nominal values in clinical settings [10], with TAVI=150, TLRI=1000, TPVARP=100, TVRP=150, TURI=400, TPVAB=50.

3.4 Random Heart Model (RHM)

To verify pacemaker software, it is essential to have a heart model that covers all possible inputs to the pacemaker. Since the pacemaker has only two input channels, and it only responds to timing relations between input actions, we use a Random Heart Model (RHM) to non-deterministically cover all pacemaker inputs. The UPPAAL model of atrial RHM is shown in Fig. 4. The interval between each action (Aget!) is a random value from the interval [*Aminwait*, *Amaxwait*]. By constraining these two parameters, the RHM covers just a subset of the heart behavior in terms of the heart rate. In the case study, we show that the single parameter representation of the heart is inadequate and argue that for complex heart conditions a more detailed heart model is required.

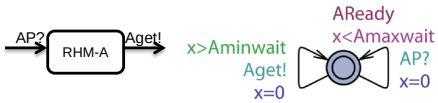


Fig. 4. Random Heart Model for Atrial Channel

4 Verification of the bottom-line safety properties

In this section, we describe the verification of the bottom-line safety properties, which need to hold for all pacemakers.

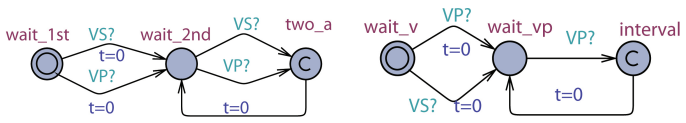


Fig. 5. (a) Monitor for LRL: Interval between two ventricular events should be less than TLRI, (b) Monitor for URL: Interval between a ventricular event and a VP should be longer than TURI

4.1 Lower Rate Limit

The most essential function for the pacemaker is to maintain the ventricular rate above a certain threshold. The monitor *Pvv* is designed in Fig. 5(a). The property *A[] (Pvv.two_a imply Pvv.t≤TLRI)* has been verified for the basic DDD pacemaker model.

4.2 Upper Rate Limit

The pacemaker can only pace the heart to increase its rate and cannot slow it down, thus it is important to guarantee it does not pace beyond a maximum rate to ensure safe operation. To this effect, an upper rate limit is specified such that the pacemaker can increase the ventricular rate up to this limit. This limit is a strict requirement that needs to hold no matter what new functions are added.

We require that a ventricle pace (VP) can only occur at least $TURI$ after a ventricle event (VS, VP). The monitor for the property is shown in Fig. 5(b) and the property *A[] (PURI_test.interval imply PURI_test.t≥TURI)* has been verified for the basic DDD pacemaker model.

4.3 Discussion

With the RHM, the heart conditions can only be encoded using the heart rate, and hence the properties are very basic. We therefore call the properties in this section *bottom-line safety properties*. We will later observe that there exist undesired behaviors even if the bottom-line safety properties are satisfied. In addition, with this simple encoding it is ambiguous whether an undesired behavior is triggered by the heart itself or by the pacemaker's intervention. In this work we are only interested in the situation when the pacemaker adversely affects the safety of the heart. In order to resolve the ambiguity, we need to adjust the heart rate so that the undesired heart condition is not covered by the heart model. When the heart condition is too complex to be encoded by the heart rate alone, a heart model with higher fidelity is needed.

5 Verification of Anti-PMT algorithms

Although the pacemaker model satisfied the bottom line safety requirements, the pacemaker can still increase the heart rate as fast as the upper rate limit, which in certain situations is inappropriate. These scenarios are referred to as Pacemaker Mediated Tachycardia (PMT). In this section, we show the existence of two well-understood PMT cases and verify whether the corresponding anti-PMT algorithms can successfully terminate the PMT.

5.1 Verification Process

The pacemaker manufacturers have developed anti-PMT algorithms to terminate different PMT scenarios. In this section, we propose a general procedure to verify the safety and correctness of such anti-PMT algorithms. The general steps for the methodology include:

1. Show existence of PMT behaviors in the closed-loop system

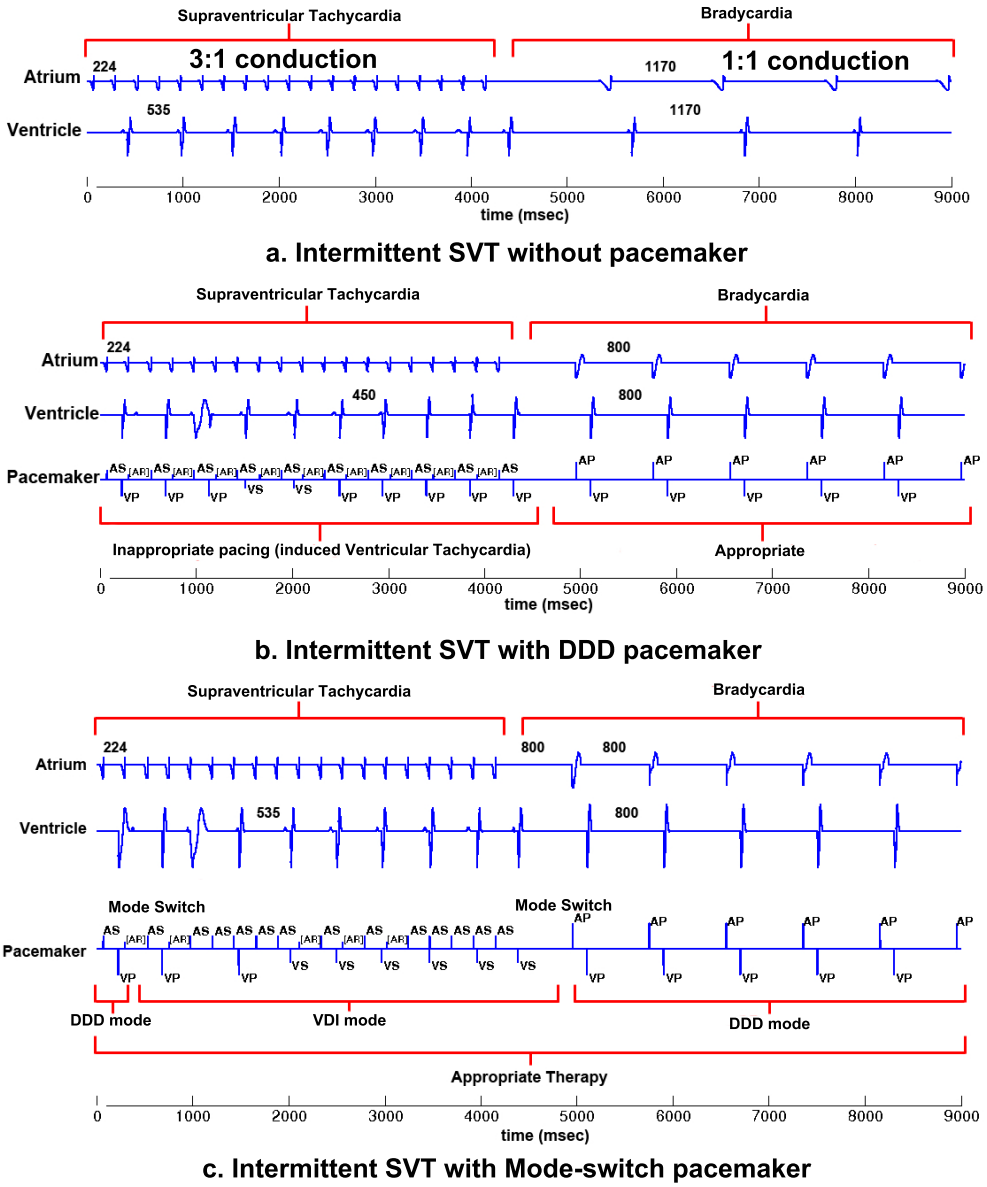


Fig. 6. Simulated Heart-Pacemaker interaction for SVT [17]

2. Introduce anti-PMT algorithms and check whether the bottom-line safety requirements still hold
3. Verify the anti-PMT algorithms by showing the non-existence of PMT scenarios

Here we use two well-identified PMT cases to demonstrate the methodology.

5.2 Verification of the Mode-Switch algorithm

Supraventricular Tachycardia (SVT): SVT is an arrhythmia which features an abnormally fast atrial rate. Fig. 6 is a series of simulation results for closed-loop interaction between a heart

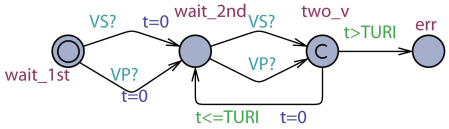


Fig. 7. Monitor for SVT: There exists an endless sequence in which interval between ventricular events is at most TURI

model with SVT and the pacemaker model. The atrial and ventricular channels show electrogram inputs to the pacemaker and the pacemaker channel shows the corresponding events received and generated by the pacemaker software, [17].

Typically the AV node, which has a long refractory period, can filter most of the fast atrial activations during SVT. This causes 2:1 or 3:1 A-V conduction and the ventricular rate remains normal (see Fig. 6 (a)). As an arrhythmia, SVT is still considered as a safe heart condition since the ventricles operate under normal rate can still maintain adequate cardiac output. However, when a dual chamber pacemaker is introduced into a heart with SVT, the AVI component of the pacemaker introduces a pathway in addition to the intrinsic conduction pathway between the atria and the ventricles. Since the atrial rate is fast, the ventricles will be paced for every sensed atrial event (AS). The pacemaker tries to maintain A-V synchrony and thus causes dangerous Ventricular Tachycardia (see Fig. 6 (b)). Since the tachycardia is caused by pacemaker, it is one case of PMT.

Mode-switch algorithms have been developed to terminate the tachycardia by switching the pacemaker into a *Fallback mode* during SVT. In the fallback mode the AVI component is disabled and the pacemaker only maintains adequate ventricular rate. Once SVT terminates, the algorithm switches the pacemaker back to dual chamber mode (see Fig. 6 (c)).

Existence of PMT during SVT: To show existence of PMT during SVT, we need to first adjust the heart model to cover SVT scenarios. The interval for the atrial RHM is set to [100, 200], so that the atrial rate is fast enough. We then need to ensure the fast ventricular rate in PMT is due to pacemaker intervention rather than the heart itself. To this end, the interval for the ventricular RHM is set to [500,800]. This rate is slow enough not to be considered as tachycardia, but faster than the Lower Rate Limit of the pacemaker so that pacemaker should not intervene. The constraint heart model covers a subset of the input space which can trigger the closed-loop system into PMT. The monitor *Pv_v* is designed to show existence of PMT during SVT. It goes to the error state if the ventricular rate drops below the Upper Rate Limit (Fig. 7).

The existence property $E[](notPv\_v.err)$, which verifies if there exists an execution in which the ventricular interval is always less or equal to TURI. The property is first verified on pacemaker without the mode-switch algorithm. The property is satisfied during verification.

Mode-Switch algorithm: Intuitively, the mode-switch algorithm first detects SVT. After confirmed detection, it switches the pacemaker from a dual-chamber mode to a single-chamber mode. During the single-chamber mode, the A-V synchrony function of the pacemaker is deactivated thus the ventricular rate is decoupled from the fast atrial rate. After the algorithm determines the end of SVT, it will switch the pacemaker back to the dual chamber mode.

The mode-switch algorithm specification we use is the same as the one used in Boston Scientific pacemakers [10]. The algorithm first measures the interval between atrial events outside the blanking period (AS, AR). The interval is considered as *fast* if it is above a threshold (*Trigger Rate*) and *slow* otherwise (see Fig. 8 (1)). A counter increments for *fast* event and decrement for *slow* event (see Fig. 8 (2)). After the counter value reaches the *Entry Count*, the algorithm will

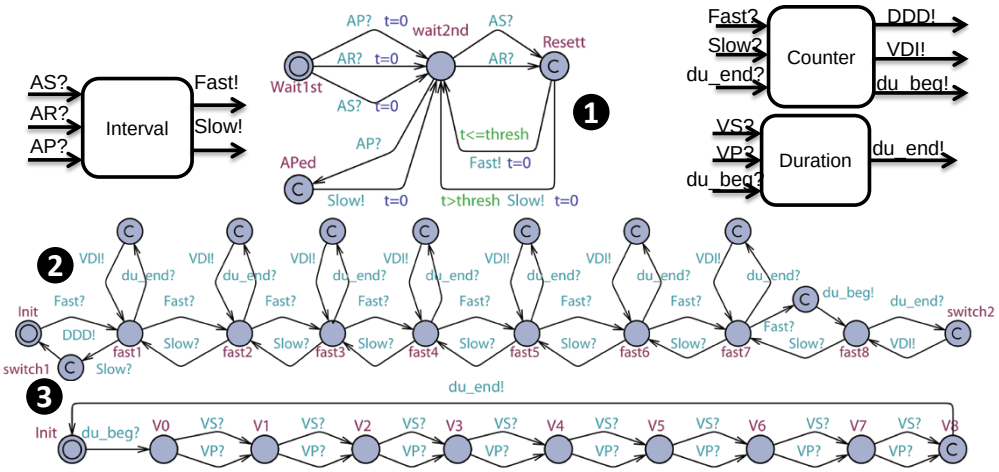


Fig. 8. Mode-Switch algorithm

start a *Duration* which is a time interval used to confirm the detection of SVT (see Fig. 8 (3)). In the *Duration*, the counter keeps counting. If the counter value is still positive after the *Duration*, the pacemaker will switch to the VDI mode (*Fallback mode*). In the VDI mode, the pacemaker only senses and paces the ventricle. At any time if the counter reaches zero, the *Duration* will terminate and the pacemaker is switched back to DDD mode.

In our UPPAAL model of the mode-switch algorithm, we use nominal parameter values from the clinical setting. We define *trigger rate* at 170bpm (350ms), *entry count* at 8, *duration* for 8 ventricular events and *fallback mode* as VDI.

In order to model both DDD and VDI modes and the switching between them, we made modifications to the AVI and LRI components. In each component two copies for both modes are modeled, and switch between each other when switching events (DDD, VDI) are received. During VDI mode, VP is delivered by the LRI component instead of the AVI component. The clock values are shared between both copies in order to preserve essential intervals even after switching.The modified AVI and LRI components are shown in Fig. 9.

Verification against bottom line safety requirements: We verify the same bottom line safety requirements on the pacemaker model with mode-switch algorithm. The Upper Rate Limit

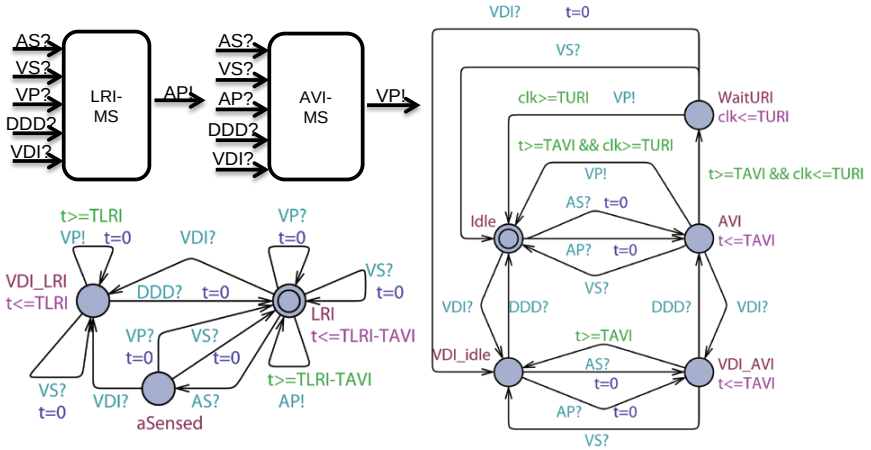


Fig. 9. New LRI & AVI component

property is satisfied but the Lower Rate Limit property is violated. When the pacemaker is switching from VDI mode to DDD mode, the responsibility to deliver VP switched from LRI component to AVI component. Since the clock reference is different in both components (Ventricular events in LRI and Atrial events in AVI), the clock value for delivering the next VP is not preserved. As a result, if an atrial event which triggered the mode-switch from VDI to DDD happens within [TLRI-TAVI, TLRI) after the last ventricular event, the next ventricular pacing will be delayed by at most TAVI time, which violates the Lower Rate Limit property (Fig. 11(a)). This safety violation should be taken into consideration by the pacemaker manufacturers.

Verification of the algorithm: We now present the verification of the correctness of the mode-switch algorithm by checking the same existence property *E[] (not Pv_v.err)* on pacemaker with mode-switch algorithm. We expect the violation of this property, since during VDI mode the ventricular rate of the heart model is less than the Upper Rate Limit and will not trigger ventricular pacing. The counter example of the violation should show that mode-switch algorithm successfully switches the mode of the pacemaker to VDI mode. However, this property is still satisfied, indicating the mode-switch algorithm failed to eliminate the PMT scenario. Since the atrial rate for our heart model is always above the trigger rate, mode switch to VDI mode will always eventually happen. The monitor PMS for the property is shown in Fig. 10. The property

Fig. 10. Monitor for Mode-Switch: Whether mode-switch to VDI mode will always eventually happen.

A<> (PMS.err) is not satisfied. The counter-example shows that some of the atrial events fall into the Post Ventricular Atrial Blanking period (PVAB) and got ignored. As a result, two *fast* intervals may be considered as one *slow* interval (see Fig. 11(b)). If this happens more than one out of the *Entry Count*, mode-switch from DDD to VDI may never happen.

Discussion: In this case study we were able to identify undesired behaviors using existence properties in UPPAAL, and showed the behavior is due to pacemaker intervention by using a constrained heart model. We also verified that the mode-switch algorithm may not always terminate undesired behavior. This scenario may not appear during open-loop testing and provides an argument in favor of verification for medical devices. The potential safety violation showed that addition of new algorithms may result in safety violations.

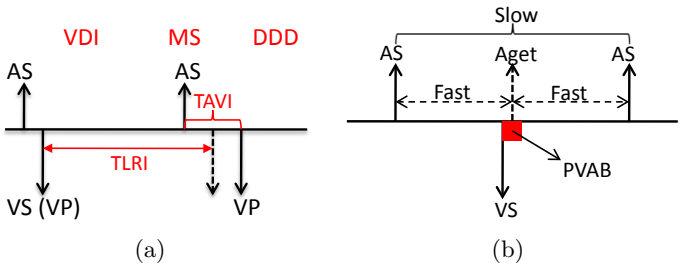
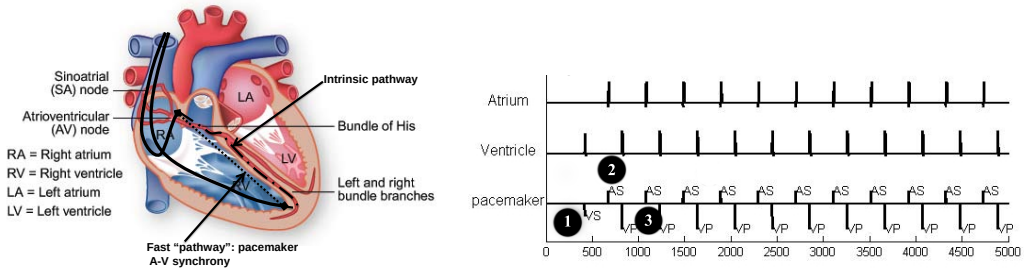


Fig. 11. (a) Safety Violation: VP is delayed due to the reset of timer during mode-switch, (b) Correctness Violation: The blocking period may block some atrial events, turning two *Fast* events to one *Slow* event



(a) Virtual circuit formed by the pacemaker and the heart

(b) EGM for ELT initialized by a early ventricular signal

Fig. 12. Endless Loop Tachycardia case study demonstrating the situation when the pacemaker drives the heart into an unsafe state [18]

5.3 Verification of Endless Loop Tachycardia (ELT) algorithm

We now focus on another PMT case, Endless Loop Tachycardia (ELT), because the mechanism of ELT is mostly independent of the heart rate. With this case study we argue that the basic RHM is inadequate to verify the correctness and safety of the anti-ELT algorithm, and hence, a more detailed heart model is required.

ELT overview: The AVI component of a dual-chamber pacemaker introduces a virtual A-V pathway which forms a loop with the intrinsic A-V conduction pathway (see Fig. 12(a)). In this scenario, a ventricular event (VS) triggers a V-A conduction through the intrinsic pathway (Marker 1 in Fig. 12(b)). The pacemaker registers this signal as an Atrial Sense (AS) (Marker 2 in Fig. 12(b)). This event triggers VP after TAVI, as if the signal conducts through the virtual A-V pathway (Marker 3 in Fig. 12(b)). The VP will trigger another V-A conduction and this VP-AS-VP-AS looping behavior will continue (see Fig. 12(b)). The interval between atrial events is TAVI plus the V-A conduction delay, which will drive the ventricular rate as high as the Upper Rate Limit.

From the pacemaker point of view, the pacemaker paces the ventricles as specified for every AS. That is why open-loop testing is unable to detect this closed-loop behavior. Modern pacemakers are equipped with anti-ELT algorithms to identify and terminate potential ELT. One common algorithm identifies ELT by the ELT pattern and terminates ELT by increasing TPVARP time once to block the AS caused by the V-A conduction.

Existence of ELT: We first ensure that the heart model does not cover the ELT pattern, so that the pattern is induced by the pacemaker. To resolve the ambiguity, we set both the

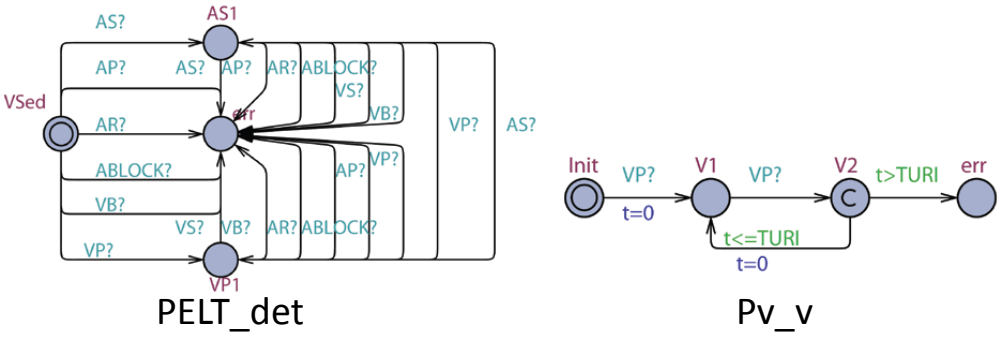


Fig. 13. Monitor for ELT: VP-AS pattern detection and Upper Rate detection

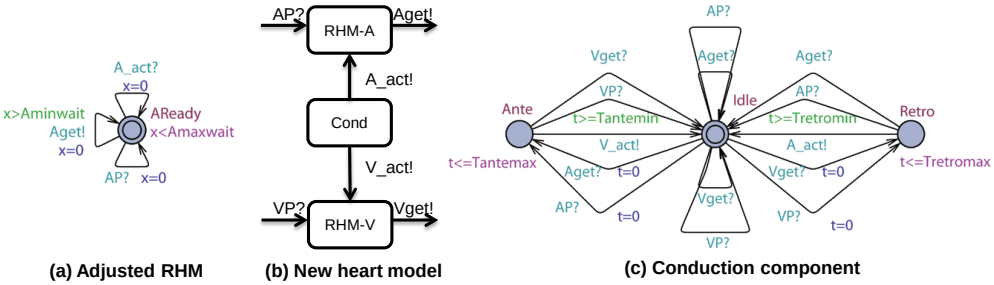


Fig. 14. Modified heart model and the conduction component

atrial interval and the ventricular interval above TURI. Two monitors were designed to show the existence of ELT. One monitor, *PELT_det*, shows the persistence of the VP-AS pattern and the other monitor, *Pvv*, shows that the ventricular rate is always no slower than the upper rate limit (Fig. 13). The existence property *E[] ((not PELT_det.err) && (not Pvv.err))* fails on pacemaker without an anti-ELT algorithm.

The reason for the failure is that the atrial rate during ELT is not determined by the atrial rate of the heart model. It is determined by the TAVI and the V-A conduction delay. This shows that describing heart conditions using the heart rate alone is inadequate.

In order to verify properties regarding ELT, we need to have a more detailed heart model. In addition to the original heart model, we model the A-V conduction of the heart. The adjusted RHM and the conduction component is shown in Fig. 14. For each atrial event *Aget*, the conduction component generates *V_act* after certain delay and vice versa. The conduction is non-deterministic so that the new heart model is an over approximation of the original one. The PVARP and VRP components are also modified to accommodate new events *A_act* and *V_act*.

After introducing the conduction component, the existence property holds, indicating the new heart model covers ELT.

The ELT-termination algorithm: The ELT detection algorithm by Boston Scientific [9] utilizes these three features:

- Ventricular rate at Upper rate limit
- VP-AS pattern
- Fixed V-A conduction delay

The pacemaker first monitors VP-AS pattern with ventricular rate at upper rate limit. Then it compares the VP-AS interval with previous intervals. ELT is confirmed if the difference between the current VP-AS interval and the first VP-AS interval are within ±32ms for 16 consecutive times. Then the pacemaker increases the PVARP period to 500ms once so that the next AS will be blocked and will not trigger a VP. ELT will then be terminated.

As the V-A conduction delays are patient-specific, the algorithm compares VP-AS interval to a previously sensed value instead of an absolute value. Since we can not store past clock values in UPPAAL, we can not explicitly model this ELT detection algorithm. However, since the conduction delay in our heart model is within a known range, we can compare the VP-AS interval with this range. The VP-AS pattern detection module for our anti-ELT algorithm is shown in Fig. 15 (1). It detects the VP-AS pattern with ventricular rate at upper rate limit and sends out *VP_AS* event if the interval qualifies.

A counter counts the number of qualified VP-AS patterns. It increases the PVARP period to 500ms if eight consecutive VP-AS patterns are detected. (Fig. 15 (2)) The PVARP component is also modified so that the PVARP period can only be changed once by the anti-ELT algorithm. (Fig. 15 (3))

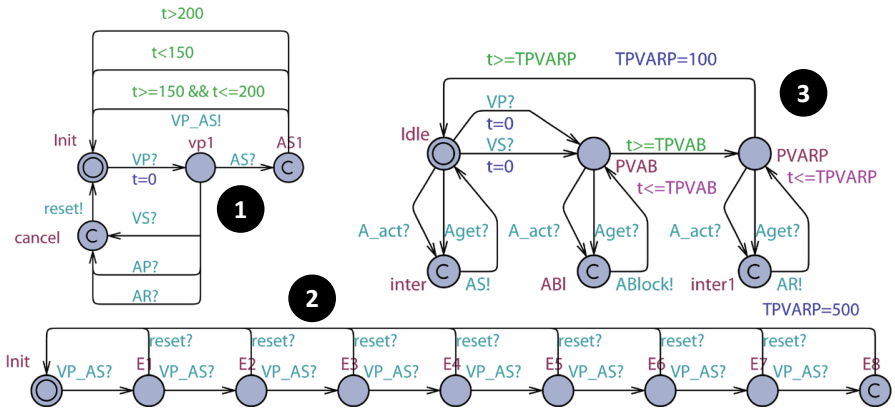


Fig. 15. Counter for VP-AS pattern

Verification against bottom line safety requirements: The two bottom line safety requirements still hold when the anti-ELT algorithm is introduced.

Verification of the algorithm: The same existence property *E[] ((not PELT_det.err) && (not Pvv.err))* is not satisfied after the anti-ELT algorithm is introduced, indicating the algorithm successfully terminates ELT. We successfully reproduced the case when the algorithm works in the simulation environment of UPPAAL.

Discussion: In this case study, we showed that a more realistic heart model is needed to specify more complex closed-loop properties. Ideally the AS event should be triggered by activation from the SA node so that it represents the intrinsic heart rhythm. However extrinsic signals such as the V-A conduction may also trigger AS. The pacemaker can not distinguish such signals from the intrinsic heart rhythm. This ambiguity often results in undesired closed-loop behaviors like ELT. As the original RHM does not provide enough information to distinguish these signals, we extended the heart model for verification of the anti-ELT algorithm.

6 Related Work

Tuan et. al propose an RTS formal model for pacemaker and its environment and verified it against number of safety properties and timed constraints using PAT model checker [19]. They have modeled the pacemaker for all 18 operating modes as described in Boston scientific, but their work lacks specifying and analyzing complex behaviors of pacemaker like mode switch.
Wiggelinkhuizen uses mCRL2 and UPPAAL to formally model the pacemaker from the firmware design of Vitatron's DA+ pacemaker [20]. Two main approaches have been used to investigate the feasibility of applying formal model checking to the design of device firmware. The main approach consists of verifying the firmware model in context of a formal heart model and a formal model of a hardware module which fails for high heart rates because of the state explosion. Another approach is to verify a part of firmware design which was feasible and was able to detect a known deadlock rather soon.
Macedo et. al have developed a concurrent and distributed real time model for a cardiac pacemaker through a pragmatic incremental approach. The models are expressed using the VDM and are validated primarily by scenario-based test, where test scenarios are defined to model interesting situations such as the absence of input pulses [21]. The models cover 8 modes of pacemaker operation.
Gomes et. al present a formal specification of pacemaker system using the Z notation in [22].

XIV

Heart Model	Pacemaker	Time Elapsed	Deadlock			LRI			URI		
			Load (states)	State Stored	State Explored	Load (states)	State Stored	State Explored	Load (states)	State Stored	State Explored
RHM w/o conduction	Basic DDD	seconds	13	229	361	9	236	366	9	251	381
RHM w/ conduction	Basic DDD	Seconds	38	1069	1785	53	1142	1897	13	1308	2148
RHM w/ conduction	DDD with anti-ELT	seconds	252	13697	24384	314	14360	25328	20	27184	41156
RHM w/ conduction	DDD with MS	1hr 30 min	65	274905	1065215	4394	76553	164992	62	521746	3857629

Fig. 16. Time Consumption and State Exploration

They have also tried to validate that the formal specification satisfies the informal requirements of Boston Scientific by using a theorem prover, ProofPower-Z. They have partially checked the consistency of their specification through reasoning. No validation experiment regarding safety conditions were performed yet. [22]
Mery et. al ,in [23], formally model all operational modes of a single electrode pacemaker system using event-B and prove them . They use an incremental proof-based approach to refine the basic abstract model of system and add more functional and timing properties. They use ProB tool to validate their models in different situations such as absence of input pulses.
Jee et. al present a safety assured development approach of real time software using pacemaker as their case study in [24]. They formally model and verify the VVI mode of pacemaker using UPPAAL and then implement it and check the preservation of properties transferred from model to implementation code.

7 Conclusion and Future Work

In this paper, we present a through analysis and verification of a dual chamber pacemaker. The specific contributions of the paper are: a) We modeled a dual chamber pacemaker in UPPAAL and verified the model against a set of safety properties. b) We identified two cases of PMT and verified the correctness and safety of two corresponding anti-PMT algorithms. c) Through this process we showed addition of more complex algorithm may result in safety violations. d) We also showed the necessity of a high fidelity heart model for verifying more complex properties. e) The procedure we used to verify anti-PMT algorithms can be used in general for medical device verification and certification. The time consumption and state exploration for the verifications are shown in Fig. 16. The UPPAAL model developed in this paper is freely available online [11]. We hope that these models can be used as a starting point for many purposes (e.g. to build models with costs and probabilities for quantitative analysis).

We only verified one instance of a DDD pacemaker since the parameters are fixed. In the future we would like to verify safety and correctness of pacemakers with parameters within the programmable range. There are also many algorithms implemented in the pacemaker to improve the quality of patient's life. As future work we would like to evaluate the efficiency of those algorithms by assigning costs for different heart conditions. The evaluation can be used to develop better treatment for general and specific patients. This requires a heart model with higher fidelity. In [7] we developed a high-fidelity heart model (VHM) which can be used for simulation and testing. The model is deterministic thus is not suitable for verification. However its timed-automata notation enables us to use model refinement to develop a series of heart model with increasing complexity. These models can be used in different levels for device certification.

References

[1] List of Device Recalls, U.S. Food and Drug Admin., (last visited Jul. 19, 2010).

[2] K. Sandler, L. Ohrstrom, L. Moy, and R. McVay. Killed by Code: Software Transparency in Implantable Medical Devices. *Software Freedom Law Center*, 2010.

[3] AUTOSAR website: www.autosar.org/.

[4] AVSI website: http://www.avsi.aero.

[5] R. Alur, D. Arney, E. L. Gunter, I. Lee, J. Lee, W. Nam, F. Pearce, S. Van Albert, and J. Zhou. Formal Specifications and Analysis of the Computer-Assisted Resuscitation Algorithm (CARA) Infusion Pump Control System. *Intl. Journal on Software Tools for Technology Transfer (STTT)*, 5:308–319, 2004.

[6] Annette ten Teije, Mar Marcos, Michel Balser, Joyce van Croonenborg, Christoph Duelli, Frank van Harmelen, Peter Lucas, Silvia Miksch, Wolfgang Reif, Kitty Rosenbrand, and Andreas Seyfang. Improving medical protocols by formal methods. *Artificial Intelligence in Medicine*, 36(3):193 – 209, 2006.

[7] Zhihao Jiang, Miroslav Pajic, and Rahul Mangharam. Cyber-Physical Modeling of Implantable Cardiac Medical Devices. *Proceeding of IEEE Special Issue on Cyber-Physical Systems*, 2011.

[8] K.G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a Nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, pages 134–152, 1997.

[9] PACEMAKER System Specification. Boston Scientific. 2007.

[10] "the compass - technical guide to boston scientific cardiac rhythm management products". 2007.

[11] Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, and Rahul Mangharam. Pacemaker UPPAAL model download: http://www.seas.upenn.edu/∼ zhihaoj/VHM/PM_verify.zip.

[12] M. Pajic, Z. Jiang, O. Sokolsky, I. Lee, and R. Mangharam. UPPAAL to Stateflow: A Model Translation Tool. Technical report, University of Pennsylvania, 2011.

[13] S. Barold, R. Stroobandt, and A. Sinnaeve. *Cardiac Pacemakers Step by Step*. Blackwell Futura, 2004.

[14] Rajeev Alur and David L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[15] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A Tutorial on Uppaal. *Formal Methods for the Design of Real-Time Systems, Lecture Notes in Computer Science*, pages 200–236, 2004.

[16] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, 1982.

[17] Z. Jiang and R. Mangharam. Modeling Cardiac Pacemaker Malfunctions with the Virtual Heart Model. *33nd Intl. Conf. IEEE Engineering in Medicine and Biology Society*, 2011.

[18] Z. Jiang, M. Pajic, and R. Mangharam. Model-based Closed-loop Testing of Implantable Pacemakers. In *ICCPS'11: ACM/IEEE 2nd Intl. Conf. on Cyber-Physical Systems*, 2011.

[19] L A. Tuan, M. C. Zheng, and Q. T. Tho. Modeling and Verification of Safety Critical Systems: A Case Study on Pacemaker. *Fourth International Conference on Secure Software Integration and Reliability Improvement*, pages 23–32, 2010.

[20] J.E. Wiggelinkhuizen. Feasibility of Formal Model Checking in the Vitatron Environment. *Master thesis, Eindhoven University of Technology*, 2007.

[21] Macedo H. D., Larsen P. G., and Fitzgerald J. Incremental Development of a Distributed Real-Time Model of a Cardiac Pacing System using VDM. *Formal Methods 2008*, pages 28–30, 2008.

[22] A. O. Gomes and M. V. Oliveira. Formal Specification of a Cardiac Pacing System. *In Proceedings of the 2nd World Congress on Formal Methods (FM '09)*, pages 692–707, 2009.

[23] D. Mery and N. K. Singh. Pacemaker's Functional Behaviors in Event-B. *Research report,INRIA*, 2009.

[24] E. Jee, S. Wang, J. K. Kim, J. Lee, O. Sokolsky, and I. Lee. A Safety-Assured Development Approach for Real-Time Software. *The Proceedings of 16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 133–142, 2010.